Computer Science 381
Programming Unix in C
The College of Saint Rose
Winter Immersion 2014

Lab 11: Advanced Pointers and Callbacks

Recommended Due Date: Thursday, January 16, 2014

In this, your final lab assignment, you will learn about some of the more advanced usage of C's pointers, and a few other items of importance that did not make it into previous labs.

Assorted Unix Fun

There are a number of Unix commands or options to commands that are well worth knowing about that have not come up so far. Here are a few to experiment with:

```
cal
units
yes
history
strings
file
cd -
pushd
popd
tail -f
du
fg, bg, kill
ps
ps -auxw
top
tee
xargs
```

Lab Question 1:

Give a brief description of each command above. Tip: try `strings` on one of your executables generated from compiling and linking a C program. (10 points)

Handling Command-Line Parameters

We have seen that you can retrieve the command line parameters to a program through the `argc` and `argv` parameters to your program's `main` function. You might have noticed that many of the Unix commands you've learned about this semester make heavy use of command-line parameters.

These often are switches that start with `-` or `--`. For example, see the man page for `ls`. You can imagine that simply parsing the command line parameters to decide what the command is supposed to do can be a complex task.

The C standard library provides a mechanism to parse these command-line parameters more easily and efficiently: `getopt`. See the description in section 3 of the manual: `getopt(3)`.

See the following example for some ways to make use of this utility.

See Example:
`/home/cs381/examples/getopt-ex`

Practice Program:

Add three new command-line options to this program. One is a required parameter specifying the name of an output file, to be specified with `-o` or `--output-file`. The second is a flag that would turn on a "debug mode" in the program by setting a variable `debug` to `1` (its default should be `0` if the flag is not specified), to be specified with `-d` or `--debug`. The last is a number which represents the enrollment in a class, is required, and must be positive, to be specified with the `-e` or `--enrollment` flags. (10 points)

Function Pointers/Callbacks

Section 5.11 of K&R shows you a way you can write a sort function to sort data in multiple ways by specifying a *function pointer* as one of the parameters to the sort function. In this case, this would be a *comparator* function that knows how to compare two values during the swap procedure. This idea is formalized in Java with the `Compatator` interface, but as usual, in C, we can achieve similar functionality but without the formal language or API support.

Make sure you understand how this works, then look at the `qsort` function provided by the C standard library (`man qsort`). This works similarly to the one in K&R, but works on an arbitrary array of any data type.

Some examples of `qsort` in action:

See Example:
`/home/cs381/examples/qsort_examples`

Practice Program:

Add functionality to the above example. Your program should be able to sort floating point values either in ascending or descending order, and should introduce the ability to read in and sort at least one new data type (a `struct` that holds some non-trivial data) by at least two different criteria. (15 points)

For another way to use function pointers in a way somewhat like an iterator, see the function `sll_visit_all` and its usage in the previous example:

See Example:
`/home/cs381/examples/sll`

Practice Program:
Add a callback function `find_max` and a call to `sll_visit_all` that demonstrates its use to the `slltest.c` program in the above example. (5 points)

Submission

Please submit all required files as email attachments to *terescoj@strose.edu*. You are recommended to do so by Thursday, January 16, 2014. Be sure to check that you have used the correct file names and that your submission matches all of the submission guidelines listed on the course home page.

Grading

Grading Breakdown	
Lab question	10 points
`getopt` enhancement	10 points
`qsort` enhancement	15 points
`find_max` enhancement	5 points